Predicting the Survivability of Breast Cancer Patients using Machine Learning

Yuning Wu ywu483@wisc.edu
Jingde Wan wan38@wisc.edu
Binhao Chen bchen276@wisc.edu

Abstract

Female breast cancer is the most prevalent and one of the most lethal cancers in the United States. In this project, we aim to train, compare and evaluate multiple state-of-the-art classification machine learning algorithms, including Logistic Regression, k-Nearest Neighbors, Decision Tree, Random Forest, XGBoost, as well as Naive Bayes to predict patients' survivability based on their physical conditions and demographic features. Before model training, we implement several feature selection and extraction techniques including Correlation Heat Map, Recursive Feature Elimination (RFE), Sequential Feature Selection (SFS) and Principal Component Analysis (PCA). We downloaded an online breast cancer patient dataset of 2509 patients with characteristics recorded in the diagnosis procedure. We also try to explore and analyze the underlying rationales behind the performance of each model. As a result, we found the Randomized Decision Tree (Extra Trees) has the best performance amongst these machine learning models we trained.

1. Introduction

Female Breast cancer is the most common type of cancer in the United States. According to the Centers for Disease Control and Prevention (CDC) data, in 2018, the rate of new cases for female breast cancer was 126.8 per 100,000 people, almost three times higher than the second-highest rate in lung and bronchus cancer. Moreover, the mortality rate of female breast cancer is ranked second behind lung and bronchus cancer.

In this project, we aim to train several classification machine learning models to predict the survivability of breast cancer patients. The dataset used in our project is the METABRIC dataset taken from The Mercurio Laboratory, where the physical conditions and characteristics of patients are obtained at the time of diagonsis[9]. We will then compare and evaluate the models and draw conclusions on their performances. The purpose of this project is to design a tool to help improve survivability prediction accuracy and limit false positive and false negative rates for future breast cancer patients' survivability prediction.

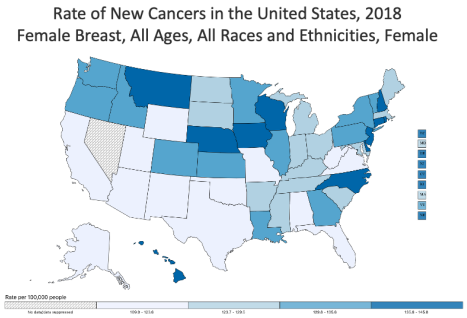


Figure 1. Rate of new Breast Cancer Case in the US. (Image Source: https://gis.cdc.gov/Cancer/USCS//AtAGlance/)

1.1. Motivation

Despite the significant amount of public awareness and medical research, breast cancer continues to grow and dominate the ranking of the type of cancer with the highest incidence among cancer-related new cases across the nation. It has been extensively researched and acknowledged by oncology professionals that the long-term survival probability of people living with breast cancer is greatly improved by early detection [5]. This is because the timely professional medical intervention will target and restrain the cancer cells in the breast and prevent it from spreading to other organs. The women victims whose breast cancer were diagnosed at the early stage would have a survival rate of 93% or higher in the first five years [1].

Machine Learning-based assistance to pathologists and physicians may be a huge leap forward in the diagnosis, prognosis, and treatment of cancer. The clinical application of machine learning in cancer diagnosis and survivability prediction might be the future of medical guidance, which can make more timely treatment plans for victims in the early stages of their condition. Ideally, if we can get a model that performs well on this data set, we hope to be able to further expand to breast cancer survivability predictions nationwide or even worldwide. However, at the same time, the acquisition of medical patient-related data and privacy

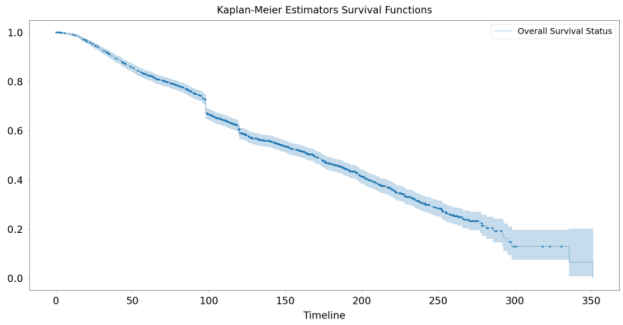


Figure 2. The Kaplan-Meier Survival Analysis curve

policies in other regions may limit the further expansion of the model.

1.2. Kaplan-Meier Analysis

Based on our breast cancer dataset, we are provided with the patients' overall survival status and overall survival months. Therefore, we are able to demonstrate the Kaplan-Meier Survival Analysis Curve, which measures the fraction of patients living for a certain amount of time after diagnosis of breast cancer. The estimation of survival function $S(t)$ is plotted as a stepwise function of the overall population of individuals of Overall Survival Status. The formula is given below:

$$\hat{S(t)} = \prod_{i:t_i<=t} (1 - \frac{d_i}{n_i})$$

where t_i is the time that event occurred, d_i is the number of death occurred at time t_i and n_i is the number of patients who are at risk but known to survive.

This curve can give us a general idea of the patients' survivability over time (Figure 2).

2. Related Work

Various Machine Learning techniques and statistical approaches have been used on breast cancer or other types of cancer patients' survivability prediction.

In 2005, Delen, Walker, and Kadam proposed a method of using artificial neural networks, decision trees, and logistic regression to develop a breast cancer prediction model for large datasets and achieve 93.6% accuracy by using C5 decision tree with 10-fold cross-validation. Their classification label of "survived" is based on the threshold of 60 months in e Survival Time Recode (STR) field. Their work provides us with great help on the preprocessing of raw data [3].

In 2008, Khan, Choi, Shin, and Kim suggested that hybrid fuzzy decision tree classification for cancer prognosis is more robust and balanced than independently applied crisp classification after using different combinations of the number of decision tree rules and types of fuzzy membership functions and inference techniques [4].

In 2010, Bellaachia and Guven found that the C4.5 decision tree algorithm on SEER Public-Use Data to predict the survivability rate of breast cancer patients achieves its best performance with an accuracy of 86.7% after several experienments[7].

In 2020, Sathipati and Ho proposed a support vector machine (SVM)-based cancer stage prediction method for HCC using an inheritable bi-objective combinatorial genetic algorithm for selecting a minimal set of miRNA biomarkers, which can maximize the accuracy of predicting the early and advanced stages of HCC[8].

However, most of the above works only focus on improving accuracy on a single machine learning model. Instead, in our project, we will emphasize evaluating different popular machine learning models' performance on the breast cancer patient dataset using different metrics with proper feature selection techniques to bring more valuable insights.

3. Proposed Method

3.1. Data Preprocessing

In the METABRIC dataset[9], there are 29 columns with missing values and only 5 columns that have no missing values. Therefore, handling missing values becomes a crucial task for us in the data preprocessing part. Our principle to deal with missing data is based on the type of the corresponding feature. For numerical features such as 'Age at Diagonis' and 'Overall Survival (Months)', the missing values are filled with the most common values in the column (based on mode) or average value, both are grouped by indicators like certain Cancer Type Detailed. For categorical features, the missing values are filled with the most common values in the column (mode) or median value, both are also grouped by indicators like certain Cancer Type Detailed. Then we need to convert all categorical features into numeric for the sake of the convenience of model training. Details of data preprocessing could be found in the Jupyter Notebook.

3.2. Feature Selection

For this project, feature selection is a notable challenge because the METABRIC dataset has over 30 features and some of them are highly correlated. A potential problem that can be caused by this is overfitting. We select features based on four methods: Correlation Heat Map, Recursive Feature Elimination (RFE), Sequential Feature Selection (SFS), and Principal Component Analysis (PCA).

3.2.1 Correlation Heat Map

Since every feature is in numeric form after conversion in the data processing stage, the correlation matrix is calculated based on the correlation of each feature and demonstrated as Correlation Heat Map in Figure 3. A larger absolute value of the correlation index indicates a stronger relationship between the two features. In this case, one of them should be dropped from the dataset. From this heatmap, 'Integrative Cluster' and 'Cellularity' are perfectly correlated so we choose to drop 'Integrative Cluster' without any loss of generality. We also notice a strong relationship between 'ER status' 'ER status measured by IHC', 'Age at Diagnosis' 'Inferred Menopausal State', 'HER2 status' 'HER2 status measured by SNP6', 'Oncotree Code' 'Tumor other Histologic Subtype'. These features will be considered to be dropped in our later discussion. Finally, rows and columns in terms of Sex are all empty because in this dataset, all breast cancer patients are biologically female so we also drop the 'Sex' column.

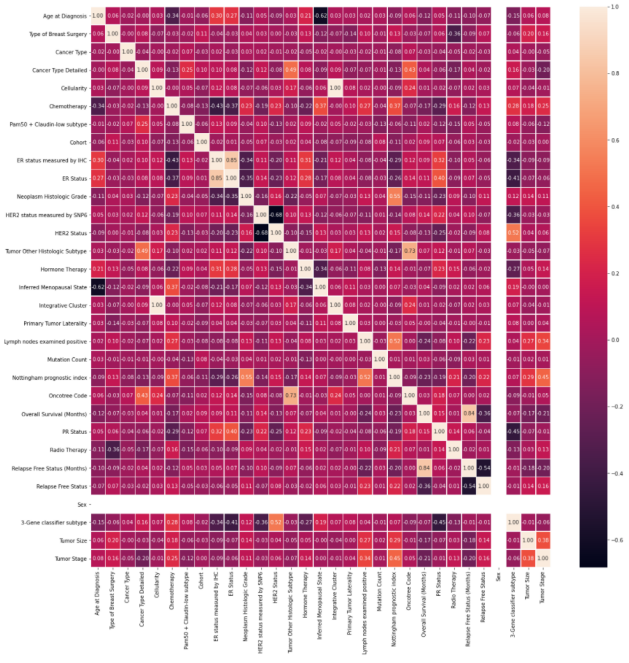


Figure 3. Correlation Heat Map

3.2.2 Recursive Feature Elimination

The goal of Recursive Feature Elimination (RFE) is to select features by recursively considering smaller and smaller sets of features until the desired number of features are attained combined with certain classification models. We implement RFE in sklearn with Random Forest models with and without cross-validation to work out the best subset of features and the number of features we need for better fitting performance. The plot showing the cross-validation score as the number of features selected is demonstrated in Figure 4. The number of features that maximize the accuracy of the machine learning model is 27. But we also notice that the second-highest cross-validation score is achieved when the number of features is 13.

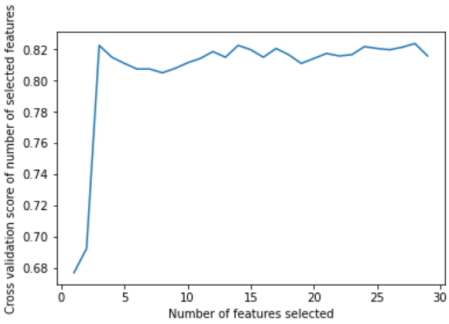


Figure 4. Recursive Feature Elimination

3.2.3 Sequential Feature Selection

We also perform the Sequential Feature Selection (SFS) by going forward. This method starts with zero feature and finds one feature that maximizes the accuracy where the certain machine learning model we trained (Here we choose k-Nearest Neighbors with n_neighbors = 5) is only based on one feature. After the first feature is selected, we repeat this process by greedily adding one more new feature until we obtain the desired number of features. We implement SFS in mlxtend with k-Nearest Neighbors model of 5 n_neighbors. The plot showing the accuracy score as the number of features selected is demonstrated in Figure 5. The number of features that maximize the accuracy of the machine learning model is 13.

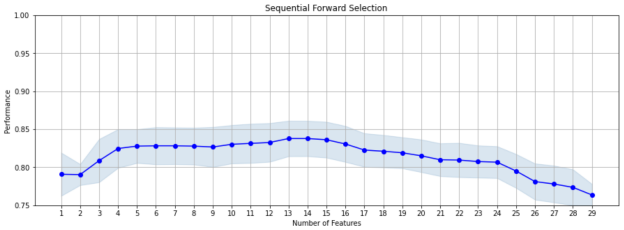


Figure 5. Sequential Feature Selection

3.2.4 Principal Component Analysis

Principal Component Analysis (PCA) is a technique to reduce the number of variables while maintaining the ma-

jority of information from features by transforming the features into lower-dimensional principal components. The principal components are linear combinations of the original variables weighted by their variance/eigenvalues in a certain orthogonal dimension. The assumption of PCA is that some variables in the dataset are linearly dependent. The plot showing the explained variance ratio as the number of principal components increases is demonstrated in Figure 6. We can observe from the plot that we need a relatively large number of principal components to explain the variance of variables in the dataset. So this dataset is not appropriate for using Principal Component Analysis as a feature extraction technique.

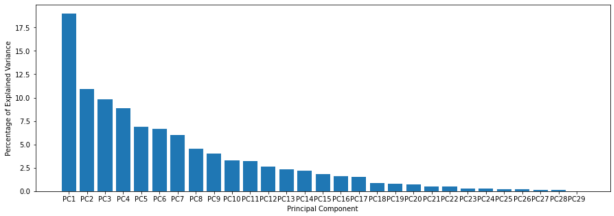


Figure 6. Principal Component Analysis

3.2.5 Summary

After implementing four feature selection and extraction techniques, we eventually decided to pick 11 out of 32 variables as features in the dataset for machine learning model training.

3.3. Machine Learning Models

We apply several classic and state-of-the-art machine learning classifiers, including ensemble methods, to our proposed dataset with an approximate $56\% - 14\% - 30\%$ split for training, validating, and testing.

3.3.1 k-Nearest Neighbors

The k-Nearest Neighbors algorithm (k-NN) is a non-parametric classification method where the objective function is only locally approximated, and all calculations (distance) are deferred until the evaluation stage. There is no explicit training step for k-NN as this algorithm literally 'memorizes' a predefined number of training samples and predicts the label from those which are closest in the distance to the new point. This algorithm computes d_i for all $i \in 1, 2, ..., N$.

$$d_i = dist\{\boldsymbol{X}^* - \boldsymbol{X}_{\text{train},i}\} \text{ for } i = 1, 2, \ldots, N$$

where $\boldsymbol{X}_{\text{train},i}$ is the features vector of the i^{th} training sample, and N is the size of the training set. Here $dist$ can be any distance metrics, e.g., standard Euclidean distance, Manhattan distance, and Minkowski distance.

3.3.2 Logistic Regression

Logistic regression, despite its name, is a linear model for classification rather than regression. It uses the logistic function to predict the conditional probability of label being 0 or 1, the standard logistic function $\sigma : \mathbb{R} \to (0, 1)$ is defined as follows:

$$\sigma(x) = \frac{e^x}{e^x + 1} = \frac{1}{1 + e^{-x}}$$

The logistic function is a sigmoid function, which outputs a value between 0 and 1. For the logit, this is interpreted as taking log-odds as input and output probability.

In this project, we apply the sklearn logistic regression package [6], which includes defaulted binary class $\ell 2$-penalized by minimizing the following cost function:

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} + C\sum_{i=1}^{n} \log\left(\exp\left(-y_i\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{X}_{\boldsymbol{i}}\right)\right) + 1\right)$$

3.3.3 Decision Tree

Besides linear models like Logistic Regression, we also consider some non-linear classifiers like the decision tree. This algorithm predicts the label of a sample by learning simple decision rules inferred from the features of training data. It can represent any binary function, the hypothesis space being searched is the entire space of binary functions. This algorithm searches for a good approximation according to the data features. As for the node split, there are several information criteria like Gini, Shannon's Entropy. Also, the maximum depth of a tree, which is also part of the hyperparameter of this model, should be predetermined before training, or it can be left as unlimited.

3.3.4 Random Forest

After trying Decision Tree, it is quite natural for us to consider the ensemble method augmented from trees by using bagging, i.e., Random Forest. Each tree in the forest ensemble is constructed by sampling with replacement (Bootstrap Sampling) from the training set. As for each node of the tree, a random set of features are chosen for splitting. These two sources of injected randomness contribute to the decrease of the variance of the ensemble model. And the prediction is collectively produced by multiple parallel decision trees, which keeps the bias of the ensemble model at a low level.

3.3.5 XGBoost

XGBoost is the abbreviation for "Extreme Gradient Boosting." It has been a very popular decision tree ensembles algorithm used to learn structured data. The model is trained in an additive manner. In each step, it fixes what have learned and greedily adds one new tree. We can represent the prediction at each step t as:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{K} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

To find the optimal tree to add in each step, we search the f_t that minimizing following objective:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t)$$

where the second term $\Omega(f_t)$ penalizes the complexity of the model and l is the loss function that measures the difference between the prediction and target [2]. Additionally, the max depth of the boosting trees needs to be tuned to avoid overfitting problem.

3.3.6 Naive Bayes

Naive Bayes methods are a collection of supervised learning algorithms based on applying Bayes' theorem with the strong assumption that every pair of features are mutually independent given the value of the label, i.e.,

$$\begin{aligned} P(x_1, \ldots, x_n) &= P(x_1 \mid y) \ldots P(x_n \mid y, x_1, \ldots, x_n - 1) \\ &= P(x_1 \mid y) \ldots P(x_n \mid y) = \prod_{i=1}^{n} P(x_i \mid y) \end{aligned}$$

Then we have:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

We applied Gaussian naive bayes model, here the likelihood of the features is assumed to follow Gaussian distribution:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Gaussian Naive Bayes aims to find parameters σ_y, μ_y that maximize $p(y) \prod_{i=1}^{n} p(x_i y)$ by using maximum likelihood estimation.

4. Experiments

4.1. Dataset

We obtain the METABRIC dataset from the Mercurio Laboratory[9]. The dataset contains the physical conditions, demographic information, as well as other characteristics of 2,509 female breast cancer patients with 837 living and 1144 deceased in terms of overall survival status. The patients are diagnosed at the age between 21.9 and 96.3. Two major surgeries are done on those patients, which are Mastectomy (removal of all breast tissue from a breast) and Breast-conserving surgery (removal of a part of the breast that has cancer). The label is the column of Overall Survival Status which indicates the survivability of breast cancer patients after treatment. It is worth noting that there are various missing values for each features so handling missing values in the dataset is crucial for this project. We split the dataset into the training set, validation set, and test set. The split is stratified by the label so that the training set, validation set, and test set all include the approximately equivalent proportion of the label.

4.2. Software

Jupyter Notebook and Python are used for code implementation and model training, where almost all of the models are trained by calling the API of scikit-learn [6]. Overleaf is used for collaborative report writing.

4.3. Hardware

Some machine learning algorithms in this project take a long time to run in PC, especially for hyperparameter tuning strategies like hyperopt. Therefore, part of the codes is run on a high-performance laptop with a strong CPU (AMD Ryzen 9 5900HX) and GPU (NVIDIA GeForce RTX 3070).

4.4. Training

Model	Hyperparameter(s)	Training Accuracy
KNN	k=50	0.834
Decision Tree (Grid Search)	max_depth=10	0.83
Decision Tree (Randomized Search)	max_depth=100, min_impurity_decrease=0.004101960004209093, min_samples_split=6	0.839
Random Forest	n_estimators=100	1
Extra Trees	n_estimators=100	1
XGBoost	alpha= 1.4199192527468378, lambda= 2.2868964529034237, learning_rate= 0.05, n_estimators= 100	0.936
Logistic Regression with Bagging	n_estimators=500	0.829
Naive Bayes		0.798

Figure 7. Experiment Summary

4.4.1 k-Nearest Neighbors

The hyperparameter tuned in the `KNeighborsClassifier` is the n_neighbors. We run the Grid Search as tuning method and examine n_neighbors k = [20, 50, 100, 150, 200]. Eventually, the GridSearch outputs k=50 as the best hyperparameter for the number of neighbors, which yields the best training and validation accuracy of 83.4% and 83.8%, respectively.

Model	95% Bootstrap Confidence Interval
KNN	[80.00, 86.35]
Decision Tree (Grid Search)	[78.21, 84.87]
Decision Tree (Randomized Search)	[76.06, 81.24]
Random Forest	[82.69, 87.65]
Extra Trees	[82.53, 87.78]
XGBoost	[82.47, 88.22]
Logistic Regression with Bagging	[80.19, 85.21]
Naive Bayes	[76.39, 82.61]

Figure 8. OOB Bootstrap 95% Confidence Interval

4.4.2 Decision Tree

For Decision Tree, we implemented two tuning techniques, namely Grid Search and Randomized Search with 10-fold cross-validation to compare and search for the best hyperparameters for the final model `DecisionTreeClassifier`. From Grid Search, we found that max_depth=10 yields the best training and accuracy, while the result by using Randomized Search shows max_depth=100, min_impurity_decrease=0.004, min_samples_split=6. Despite the major difference in max_depth by using these two tuning methods, the training accuracy is very close with Grid Search at 83.0% and Randomized Search at 83.9% as shown in the table in Figure 7.

4.4.3 Random Forest & Extra Tree

The max depth of each tree, the max size of random features, as well as the maximum number of trees of the forest, are part of the hyperparameters that can be predetermined before finalizing the `RandomForestClassifier` and `ExtraTreesClassifier`. For Random Forest, by using Grid Search with 5-fold cross-validation, we finally end up with max_depth = 90, max_features = 5, n_estimators = 100. As for the randomized decision trees model, we directly use the default hyperparameter to train and obtain an unexpected superb performance. The high accuracy could be attributed to the two randomnesses of various sub-samples and the randomized subset of features, which makes this model be able to improve the predictive accuracy and control over-fitting.

4.4.4 XGBoost

We performed hyperparameter tuning on `XGBClassifier` by using hyperopt with 10-fold cross-validation. The hyperparameters are n_estimators, learning_rate, lambda, and alpha. To maximize the average accuracy, we eventually end up with n_estimators = 100, learning_rate = 0.05, lambda = 1.98 and alpha = 1.04, which yields the best training and validation accuracy of 94.4% and 86.1% respectively.

4.4.5 Logistic Regression

Besides experimenting solely on the Logistic Regression model, we also try ensemble methods Bagging and AdaBoosting with `LogisticRegression` as the base-estimator. On this training set, bagging has a better fitting performance in terms of train/test accuracy. We trained the ensemble method with varying sizes of bagging from 10 to 100, the test accuracy of the ensemble model fluctuated but it achieve higher accuracy compared to a sole logistic regression model when the bagging size is greater than 50.

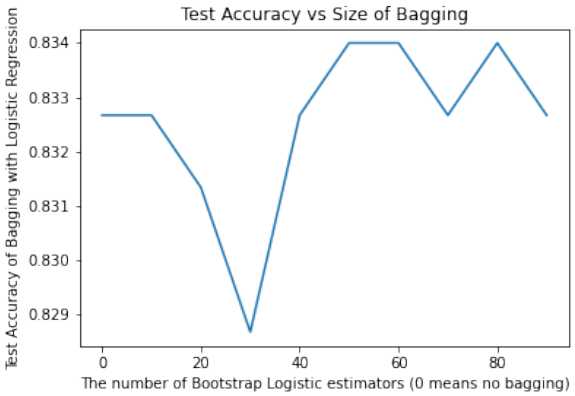


Figure 9. Logistics Regression Accuracy v.s. The Size of Bagging

4.4.6 Naive Bayes

The parameters that are trained in Naive Bayes are the prior probabilities of different class labels, as well as the likelihood of different features for each class. Naive Bayes has almost no hyperparameters to tune, it usually generalizes well. However, it is the only classifier whose test accuracy is below 80% among all the models we attempted to train. We suspect this is because our data set does not satisfy the strongest assumption of Naive Bayes: Each feature makes an independent and identical contribution to the class label prediction. This is a strong assumption and unrealistic for real data.

5. Results and Discussion

In evaluating the generalization performance of our models, we computed test accuracy, precision, recall, F1, and MCC for each model along with 2×2 confusion matrices as shown in Figures 10 and 11.

By observing accuracy alone, our Extra Trees model has the best performance at 0.861 testing accuracy, followed

Model	Test Accuracy	Precision	Recall	F1	MCC
KNN	0.838	0.877	0.811	0.843	0.678
Decision Tree (Grid Search)	0.807	0.838	0.794	0.815	0.615
Decision Tree (Randomized Search)	0.837	0.827	0.878	0.852	0.672
Random Forest	0.846	0.875	0.831	0.852	0.693
Extra Trees	0.861	0.882	0.854	0.868	0.721
XGBoost	0.858	0.881	0.849	0.865	0.716
Logistic Regression with Bagging	0.833	0.862	0.819	0.84	0.666
Naive Bayes	0.798	0.828	0.787	0.807	0.597

Figure 10. Model Evaluation Summary

closely by XGBoost at 0.858, and Random Forest at 0.846. Meanwhile, Naive Bayes, Decision Tree (Grid Search), and Logistic Regression with Bagging appear to have the lowest accuracy.

However, it is insufficient to look at accuracy alone, especially in the context of our study (Binary Classification) where both a Type I (high false negative rate) and a Type II (high false positive rate) error have severer impacts than many other studies. In this case, a Type I error occurs when the model falsely predicts that a patient will decease and Type II error occurs when the model falsely predicts that a patient will survive.

In the context of our study, a Type II error will lead to much more dire consequences than a Type I error. The reason is that falsely predicting that a patient will survive–underestimating the severity of their current conditions–can result in less medical attention, and lighter medication doses, etc. Therefore, in evaluating the performance of our models, we prioritized minimizing the Type II error. In practice, we seek a higher recall rate since it signifies a lower rate of type II error occurrence.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1\ Score = 2\,\frac{Recall \cdot Precision}{Recall + Precision}$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}$$

where

TP is True Positives, also known as correctly predicted positive values;

TN is True Negatives, also known as correctly predicted negative values;

FP is False Positives, when actual class is no and predicted class is yes;

FN is False Negatives, when actual class is yes but predicted class in no.

As shown above, when True Positive rates are constant, recall increases as the False Negative number decreases.

As Figure 10 shows, Decision Tree using Randomized Search resulted in the best recall score at 0.878, followed by Extra Trees at 0.854, and XGBoost at 0.849. These results are largely consistent with test accuracy, with only Decision Tree replacing Random Forest in the top 3. The bottom performers are Naive Bayes, Decision Tree (Grid Search), k-Nearest Neighbors, and Logistic Regression with Bagging, which is consistent with the test accuracy results as well.

Figure 11 provides a more detailed view into the False Negative rates of each model using Confusion Matrices which is also consistent with our findings summarized in Figure 10.

We concluded that Extra Trees have the best model performance out of all models we have tested. It has the best test accuracy, precision, F1 and MCC scores. Despite that its recall score being lower than Decision Tree with Randomized Search, it is still the second-highest among all models.

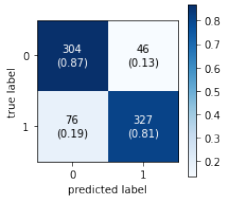
6. Conclusions

This project aims to search for the most right machine learning model (with limitations explained below) for predicting breast cancer patients' survivability. We trained, evaluated, compared, and analyzed eight machine learning models, namely k-Nearest Neighbors, Decision Tree (Grid Search), Decision Tree (Randomized Search), Random Forest, Extra Trees, XGBoost, Logistic Regression with Bagging, and Naive Bayes with multiple tuning techniques. In the end, we discovered Extra Trees to be the best performing model in terms of test accuracy as well as the recall rate.

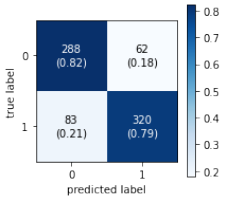
Despite the limitations of using a relatively small dataset (2509 entries), our implementation using Extra Trees achieves a good performance at a 0.861 test accuracy and equally good precision and recall scores.

It is worth noting that we did not use any tuning technique such as Grid Search to find the best hyperparameters for the Extra Tree model, instead, we trained with default parameters for the sake of computational efficiencies–due to the limitations of using local CPU instead of cloud computing. For future studies, one can implement more tuning techniques on the Extra Tree model to better the model performance. One can also implement XGBoost models with different hyperparameters as a comparison since it is the second-best model in our study.

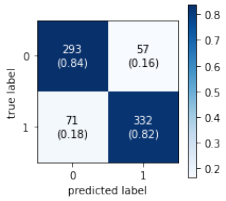
Finally, we hope the machine learning algorithm will be useful and reliable in predicting breast cancer patients' survivability with our model and further hyperparameters tuning and aid medical professionals in making better treatment decisions.



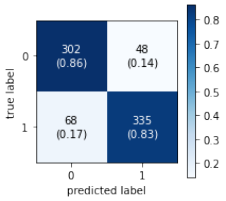
(a) k-Nearest Neighbors



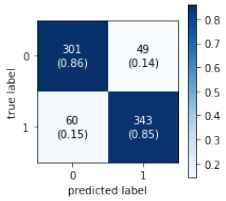
(b) Decision Tree (Grid Search)



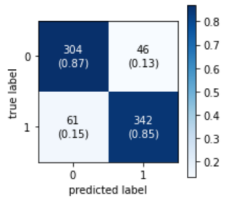
(c) Decision Tree (Randomized Search)



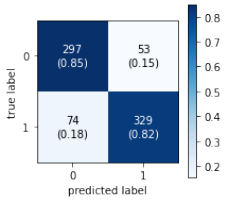
(d) Random Forest



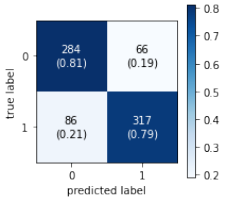
(e) Extra Trees



(f) XGBoost



(g) Logistic Regression with Bagging



(h) Naive Bayes

Figure 11. Confusion Matrices

7. Acknowledgements

We want to thank Professor Sebastian Raschka for his wonderful lectures as well as timely help on piazza and office hours. We are also grateful for the METABRIC dataset organized and uploaded by user Gunes Evitan on Kaggle.

8. Contributions

Binhao Chen discovered the dataset online and contributed to the model selection, training, as well as the tuning part of the code, which are summarized in section 3.3 and section 4(Experiments) of the written report. Jingde Wan contributed mainly in the data cleaning, pre-processing and feature selection for future analysis and model fitting, which correspond to section 3.1 and 3.2 of this written report. Jingde also drafted the Abstract, Introduction(section 1) and Related Work(section 2) section of this report. Yuning Wu contributed to the model evaluation and result visualization, which correspond to Results and Discussion(section 5) of the report. Yuning Wu also finished the Conclusion (section 6), Acknowledgements(section 7) of the report.

References

[1] Early detection is key. Available at `https://www.carolmilgardbreastcenter.org/early-detection`.

[2] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[3] D. Delen, G. Walker, and A. Kadam. Predicting breast cancer survivability: a comparison of three data mining methods. *Artificial Intelligence in Medicine*, 34(2):113–127, 2005.

[4] U. Khan, J. Choi, H. Shin, and M. Kim. Predicting breast cancer survivability using fuzzy decision trees for personalized healthcare. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2008:5148–51, 02 2008.

[5] A. M. Leitch, G. D. Dodd, M. Costanza, M. Linver, P. Pressman, L. McGinnis, and R. A. Smith. American cancer society guidelines for the early detection of breast cancer: update 1997. *CA: A cancer Journal for Clinicians*, 47(3):150–153, 1997.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] A. S. Sarvestani, A. A. Safavi, N. Parandeh, and M. Salehi. Predicting breast cancer survivability using data mining techniques. In *2010 2nd International Conference on Software Technology and Engineering*, volume 2, pages V2–227–V2–231, 2010.

[8] S. Y. Sathipati and S.-Y. Ho. Novel mirna signature for predicting the stage of hepatocellular carcinoma. *Scientific Reports*, 10, 2020.

[9] The Mercurio Laboratory. The metabric dataset. Available at `https://www.kaggle.com/gunesevitan/breast-cancer-metabric`, 2012.